Texture Mapping

(Some Images from Rosalee Wolff)
Texture Mapping

• Problem with shading models:
  • They assume that a diffuse surface has uniform reflectance.
• This is okay for walls or solid balls, but not most objects
• We could add geometric complexity
  • This is too time consuming.
• Alternative: Texture mapping
  • Developed by Catmull (1974), Blinn and Newell (1976), and others
Texture Mapping

Goal: add visual detail without adding geometric detail

8 polygons 8 polygons
Typical Textures
Repeating Textures
Repeating Textures
Textures

Any image can be used as a texture map
Texture Mapping Applet
Texture Mapping

• There are both 2D and 3D versions of texture mapping
• 2D – “wallpaper” a 2D image onto an object
• 3D – “carve” a 3D object out of a block
2D Texture Mapping

• Given a 2D texture (image) and a 3D object, map the texture onto the object.
• Where does each point on the object map into the texture?
Texture Map Shapes

• Planar Map
  • Simply remove one of the object’s coordinates (project onto that coordinate plane)
Planar Map

- The texture is constant in one direction.
  - Often not what you want.
  - E.g., projecting along Z
Planar Map

• Projecting along X and Y instead
Cylindrical Map

• We could use a cylindrical map instead.
• \((x, y, z)\) is converted to \((r, \theta, \text{height})\).
• For texture mapping, \(\theta\) is converted into a \(u\)-coordinate and \(\text{height}\) is converted into a \(v\)-coordinate.
• This wraps the texture map around the object.
Cylindrical Map

• At minimum and maximum extents of the cylinder, the texture gets pinched together.
• E.g., a cylindrical map parallel to Z:
Cylindrical Map

- Similarly if the map is parallel to X or Y:
Spherical Map

• Convert from (x, y, z) to spherical coordinates.
• Latitude is converted to a \textit{u-coordinate}, longitude is converted to a \textit{v-coordinate}.
Spherical Map

• This still pinches the texture at the poles, but it’s different from using a cylindrical map.
Spherical Map

• With the poles in the X and Y directions:
Box Map

• We can use a collection of planar maps to provide better coverage than using a single planar map:
Box Map

- We project each plane onto its portion of the object.
- Use the object’s normal to determine which texture to use.
Box Map

• This produces:
Box Map

• Using different textures:
Box Map

• We get:
Map Entity

• To texture map, we take an \((x, y, z)\) value from the object and determine a \((u, v)\) texture value.

• How we determine what we use as the \((x, y, z)\) value is the *map entity*.

• Can use various things:
  - the position relative to the object’s bounding box
  - the surface normal at the point
  - a vector running from the object’s center through the point
  - the reflection vector at the current point
Map Entity
Map Entity

• Using the same map shape, but different map entity can give quite different results.
• Planar Mapping:
Map Entity

- Cylindrical Mapping:
Map Entity

- Spherical Mapping:
Map Entity

• Box Mapping:
Another Example
The Texture Mapping Process

Texture Map

Surface

Pixel

(inverse map)

(raytrace)
• Consider surface visible at current pixel.
• Find the patch on the surface that corresponds to it.
  • Map screen coord of pixel corners back to object
  • Find texels that map to the surface patch
  • If multiple texels lie on patch combine them:
    • weighted average or supersampling
Mapping Parametric Surfaces

- Parametric surfaces are already parameterized by (s, t).
- Use the (s,t) parameters as the (u,v) texture parameters.
The Utah Teapot

- 32 Parametric patches
If we assign values to the vertices in the range (0,1), we can use the same texture mapping approach:
Non-linear Mapping

• We can distort the texture using a non-linear mapping:
3D Mapping – Solid Textures
**Triangles**

- Given a triangle defined by three points \((a, b, c)\), how do we associate a texture color with a point on the triangle?
Computing the Point

• Given the \((x,y)\) point in the triangle, how do we transform that to a \((u,v)\) point in the image?
• Set up a non-orthogonal coordinate system with origin \(a\) and basis vectors \(b-a\) and \(c-a\)
Barycentric coordinates

- Any point on the triangle can be defined by the barycentric coordinate

\[ p = a + \beta(b-a) + \gamma(c-a) \]
Barycentric coordinates

• Once we have computed the \((\beta, \gamma)\) barycentric coordinate for the triangle, we can determine the corresponding \((u, v)\) point.

• First, establish the \((u, v)\) system:
Computing the \((u, v)\) coordinate

\[
\begin{align*}
    u(\beta, \gamma) &= u_a + \beta(u_b - u_a) + \gamma(u_c - u_a) \\
    v(\beta, \gamma) &= v_a + \beta(v_b - v_a) + \gamma(v_c - v_a)
\end{align*}
\]
Performing the transformation

- We can then compute the color for that point of the triangle.
Textures in OpenGL

- OpenGL provides routines for texture mapping
- To use, you must:
  - Define the texture
  - Specify the manner in which the texture is to be applied to the pixels.
  - Enable texture mapping.
  - Draw the objects, including both geometric and texture coordinates.
Defining the Textures

• “Bind” the texture to a particular id
• The id is how you refer to the texture later
  `glBindTexture(GL_TEXTURE_2D, 3);`
• When you need to access the texture later, use the id
Texture Format

• You must also tell OpenGL the format to read the data in

```c
glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
```

• This specifies that the data has one byte for red, one byte for green, and one byte for blue

• This is probably the way your data is stored
Setting Texture Parameters

- OpenGL supplies a routine to set various parameters:
  
  ```
  glTexParameteriv(target, pname, param);
  ```

- Where
  
  - `target` is `GL_TEXTURE_2D`
  - `pname` is a parameter name that you want to change:
    - `GL_TEXTURE_WRAP_T`
    - `GL_TEXTURE_WRAP_S`
    - `GL_TEXTURE_MIN_FILTER`
    - `GL_TEXTURE_MAX_FILTER`
  - `param` is the parameter value to set to
Setting Texture Parameters

• You probably want:

```c
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```
Original Texture

Repeated in both s and t

Clamped in both s and t
Setting Up the Texture Environment

• You need to tell OpenGL how to treat textures in the environment

\[
glTexEnvf(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, param);
\]

• Where param is:
  • GL_MODULATE – uses environment lighting
  • GL_DECAL – does not use environment lighting
  • GL_BLEND – blends the object color with environment color
  • GL_REPLACE – just uses object color
Preparing the Texture

- With all of the parameters set, we can now use the texture:

  `glTexImage2D(target, level, internalformat, width, height, border, format, type, pixels)`

- With:
  - `target`: `GL_TEXTURE_2D`
  - `level`: level of detail number, you probably want 0
  - `internalformat`: the number of color components in the texture, you probably want `GL_RGB`
  - `width`: width of image, must be 2^n + 2b, where n is some number, and b is the size of the border
  - `height`: height of the image, must be 2^m + 2b, where m is some number, and b is the size of the border
  - `border`: width of the border, must be 0 or 1
  - `format`: the format of the pixel data, you probably want `GL_RGB`
  - `type`: the data type of the pixel data, such as `GL_UNSIGNED_BYTE`, `GL_FLOAT`, etc.
  - `pixels`: a pointer to the image in memory
Using the Texture

• Once the `glTexImage2D` call is made, you can then use the texture
  • Enable texturing using
    ```
    glEnable(GL_TEXTURE_2D)
    ```
  • Make sure that you bind the texture before a `glBegin/glEnd` pair
  • Specify a texture coordinate for each vertex coordinate
    ```
    glTexCoord2f(u, v);
    glVertex3f(x, y, z);
    ```
Texture Mapping Example

```c
BindTexture (GL_TEXTURE_2D, 13);
Begin (GL_QUADS);
TexCoord2f (0.0, 0.0);
Vertex3f (0.0, 0.0, 0.0);
TexCoord2f (1.0, 0.0);
Vertex3f (10.0, 0.0, 0.0);
TexCoord2f (1.0, 1.0);
Vertex3f (10.0, 10.0, 0.0);
TexCoord2f (1.0, 1.0);
Vertex3f (10.0, 10.0, 0.0);
End () ;
```
Texture IDs

• You can have the system generate unique IDs for you so that you don’t have to do it manually.

  glGenTextures(n, textures)

• Where:
  • n is the number of texture IDs you want generated
  • textures is an array in which to store the IDs
Pseudo Code for Texture Mapping

```c
void setupOpenGL (void)
{
    glEnable (GL_TEXTURE_2D);
}
void loadAllTextures (void)
{
    glBindTexture (... , 1);
    glPixelStorei (...);
    glTexParameteri (...);
    glTexEnvf (...);
    glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB, imageWidth, imageHeight, 0,
                 GL_RGB, GL_UNSIGNED_BYTE, imageData);

    glBindTexture (... , 2);
    glPixelStorei (...);
    glTexParameteri (...);
    glTexEnvf (...);
    glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB, imageWidth2, imageHeight2, 0,
                 GL_RGB, GL_UNSIGNED_BYTE, imageData2);

    ...
}
```
Pseudo Code for Texture Mapping

```c
void drawTextureObjects (void)
{
    glBindTexture (... , 1);
    glBegin (...);
        glTexCoord (...);
        glVertex (...);
    glEnd (...);

    glBindTexture (... , 2);
    glBegin (...);
        glTexCoord (...);
        glVertex (...);
    glEnd (...);
    ...
}
```
Other Mapping Applications

• Color: diffuse component of surface (standard texture mapping)
• Reflection: specular component of surface to simulate reflection (environment mapping)
• Normal vector: simulate 3D surface structure (bump mapping)
• Geometry: raise/lower points to actually modify surface (displacement mapping)
• Transparency: vary the transparency across the object
Environment / Reflection Mapping

- Texture contains an image of the surroundings.
- Map the reflection direction.
- This is an approximation.
  - Less accurate, but more efficient than ray-tracing.
  - Does not compute self-reflections.
Example
Transparency

Lots of Bump Mapping

Volumetric Lighting

Sky Texture

Transparency
Texture, Bump, Specular
Self-Illumination, Transparency
Sky Texture
Self-Illumination
Texture, Bump, Specular
Self-Illumination, Transparency
Comparison

Bump Mapping

Horizon Mapping (shadows)

Displacement Mapping

View Dependent Displacement Mapping